# MANAGING SECURITY ACROSS MULTIPLE ENVIRONMENTS WITH DEVSECOPS

## PHASE 3- SOLUTION DEVELOPMENT AND TESTING

**College Name:** KNS Institute of Technology

**Members:**

- **Name : Mohammed Thayeeb Shariff [ Team Lead]**

  USN : 1KN21IS031 [ EMAIL ]

  CAN ID : 32887773
- **Name : Nidith V S**

  USN : 1KN21IS038

  CAN ID : 32888579
- **Name : Ramachandragowda S Patil**

  USN : 1KN21CS082

  CAN ID : 32888557
- **Name : Sateesh Biradar**

  USN : 1KN21IS041

  CAN ID : 32889102

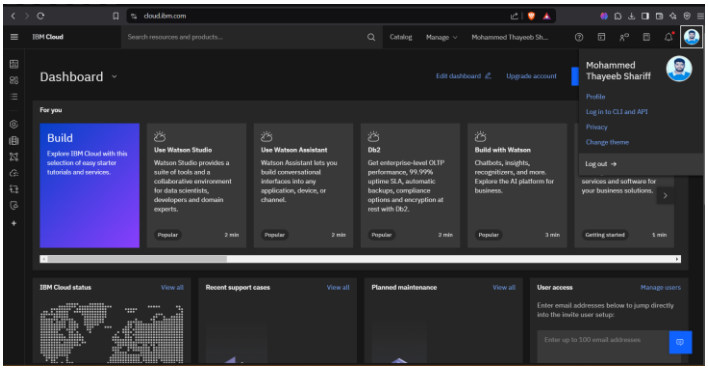
---

### SECTION 1: SOLUTION DEVELOPMENT

### Setting up IBM Cloud Environment and Configuring Necessary Tools

### Step 1: Create an IBM Cloud Account

1. Navigate to IBM Cloud.
2. Sign up for an account (or log in if you already have one).
3. Ensure you have a billing account set up to access IBM Cloud services.



**Step 2: Install Required Tools Locally**

1. **Install Minikube**:
    - Follow the instructions from the Minikube installation guide.
2. **Install kubectl**:
    - Download and set up the kubectl CLI using the official guide.
3. **Install Docker**:
    - Set up Docker for building and managing container images (Docker installation guide).

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  COMMENTS

tayyab@Tayyab:/mnt/c/Users/Tayyab Qadri/OneDrive/Desktop/final$ minikube version
minikube version: v1.34.0
commit: 210b148df93a80eb872ecbeb7e35281b3c582c61
tayyab@Tayyab:/mnt/c/Users/Tayyab Qadri/OneDrive/Desktop/final$ kubectl version --client
Client Version: v1.31.0
Kustomize Version: v5.4.2
tayyab@Tayyab:/mnt/c/Users/Tayyab Qadri/OneDrive/Desktop/final$ docker --version
Docker version 27.4.1, build b9d17ea
tayyab@Tayyab:/mnt/c/Users/Tayyab Qadri/OneDrive/Desktop/final$
```

**Step 3: Set Up IBM Cloud Container Registry**

**Before starting this make sure u have Logged in to IBM Cloud Via Linux Environment.**

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  COMMENTS

ibmcloud: command not found
tayyab@Tayyab:/mnt/c/Users/Tayyab Qadri/OneDrive/Desktop/final$ curl -fsSL https://clis.cloud.ibm.com/install/linux

Current platform is linux64. Downloading corresponding IBM Cloud CLI...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 10.0M  100 10.0M    0     0  5404k      0  0:00:01  0:00:01 --:--:-- 5406k
Download complete. Executing installer...
Bluemix_CLI/
Bluemix_CLI/install_bluemix_cli
Bluemix_CLI/autocomplete/
Bluemix_CLI/autocomplete/zsh_autocomplete
Bluemix_CLI/autocomplete/bash_autocomplete
Bluemix_CLI/bin/
Bluemix_CLI/bin/ibmcloud
Bluemix_CLI/bin/LICENSE
Bluemix_CLI/bin/ibmcloud.sig
Bluemix_CLI/bin/NOTICE
Bluemix_CLI/install
Bluemix_CLI/uninstall
Superuser privileges are required to run this script.
[sudo] password for tayyab:
Sorry, try again.
[sudo] password for tayyab:
Install complete.
tayyab@Tayyab:/mnt/c/Users/Tayyab Qadri/OneDrive/Desktop/final$ ibmcloud --version
ibmcloud 2.32.2 (c23867a-2025-02-06T19:48:11+00:00)
Copyright IBM Corp. 2014, 2025
tayyab@Tayyab:/mnt/c/Users/Tayyab Qadri/OneDrive/Desktop/final$ ibmcloud login
API endpoint: https://cloud.ibm.com
```

**Try to Login utilizing the LOGIN IBM CLI feature.**

**ibmcloud login -a https://cloud.ibm.com -u passcode -p <your passcode>**

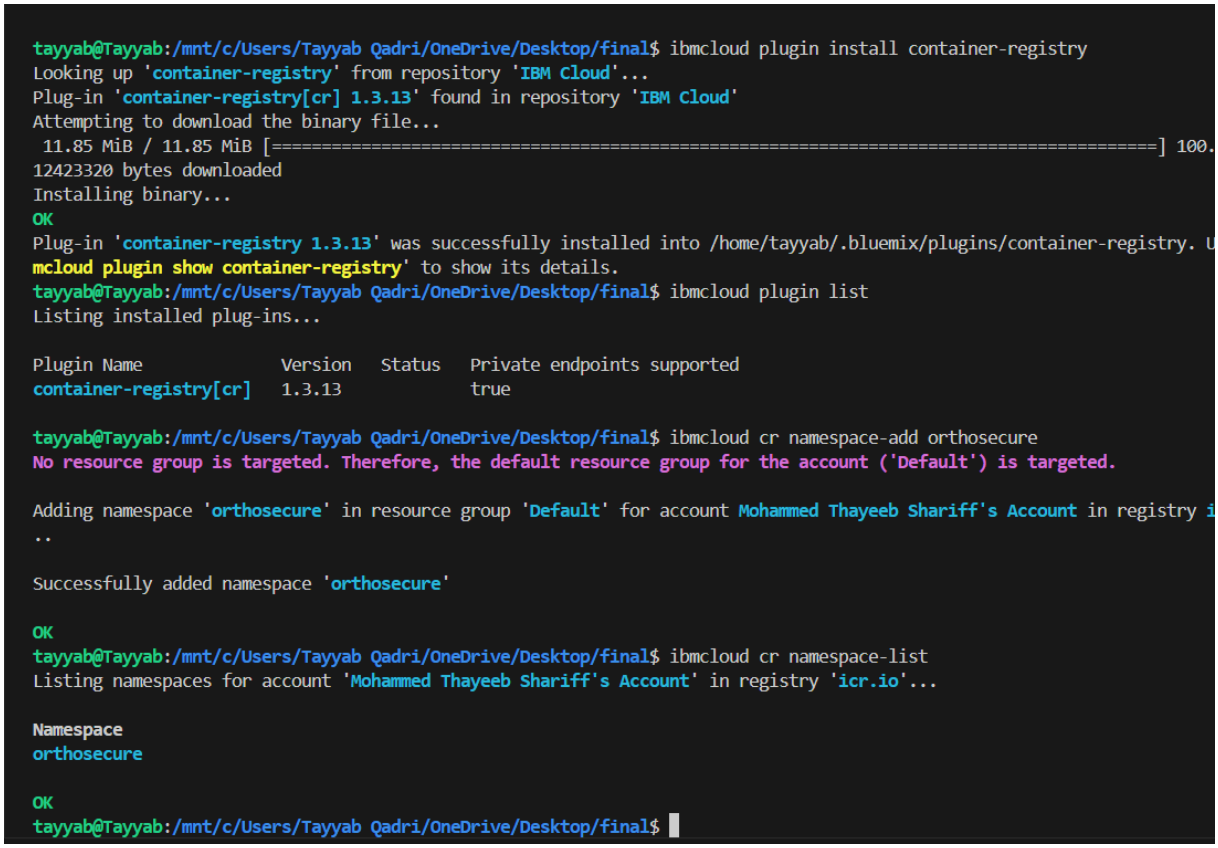
```
tayyab@Tayyab:/mnt/c/Users/Tayyab Qadri/OneDrive/Desktop/final$ ibmcloud login -a https://cloud.ibm.com -u passcode -p H3nLKf
7IGm
API endpoint: https://cloud.ibm.com
Authenticating...
OK

Targeted account Mohammed Thayeeb Shariff's Account (63573f9b53aa41fc880b7129ad615f83)


Select a region (or press enter to skip):
1. au-syd
2. in-che
3. jp-osa
4. jp-tok
5. eu-de
6. eu-es
7. eu-gb
8. ca-tor
9. us-south
10. us-east
11. br-sao
Enter a number>


API endpoint:     https://cloud.ibm.com
Region:
User:             devgenius9211@gmail.com
Account:          Mohammed Thayeeb Shariff's Account (63573f9b53aa41fc880b7129ad615f83)
Resource group:   No resource group targeted, use 'ibmcloud target -g RESOURCE_GROUP'
tayyab@Tayyab:/mnt/c/Users/Tayyab Qadri/OneDrive/Desktop/final$
```

1. From the IBM Cloud dashboard, search for "Container Registry." Plugin wont be available in Linux so install it

2. Create a namespace for your container images:
   ibmcloud cr namespace-add <namespace_name>
   In our case --- > **ibmcloud cr namespace-add orthosecure**

3. Enable image vulnerability scanning:

   **ibmcloud cr policy-update --scan-on-push true**




**Implementing Containerization and Pushing to IBM Cloud Container Registry**

**Step 1: Dockerize the Application**

1. **Create Dockerfiles**:

**Frontend Dockerfile** (/app/Dockerfile):

```
FROM python:3.11-slim
# Prevent .pyc files and enable unbuffered output
ENV PYTHONDONTWRITEBYTECODE=1
ENV PYTHONUNBUFFERED=1
# Install system dependencies, including MySQL
client
RUN apt-get update && apt-get install -y \
    gcc \
    libpq-dev \
    default-mysql-client \
    && rm -rf /var/lib/apt/lists/*
# Set working directory
WORKDIR /app
# Copy dependencies first to leverage Docker cache
COPY requirements.txt /app/
RUN pip install --upgrade pip
RUN pip install --no-cache-dir -r requirements.txt
# Copy application files
COPY . .
# Add wait-for script
```

```
COPY wait-for.sh /wait-for.sh
RUN chmod +x /wait-for.sh
# Set Python path for the app
ENV PYTHONPATH=/app
# Expose application port
EXPOSE 5000
# Default command
CMD ["python", "main.py"]
```

**Wait-for.sh File ----- >**

```
#!/bin/bash
host="$1"
shift
cmd="$@"
echo "Waiting for MySQL ($host)..."
until mysqladmin ping -h "$host" --silent; do
  echo "MySQL is unavailable - waiting..."
  sleep 2
done
echo "MySQL is up - executing command"
exec $cmd
```

**Requirements.txt File ---- >**

1. hvac
2. flask
3. typing
4. python-dotenv
5. dnspython
6. flask
7. mysql.connector
8. werkzeug
9. flask_wtf
10. wtforms
11. flask_mail
12. setuptools
13. pathlib
14. gunicorn
15. pytest

- **Backend Dockerfile** (/Db/Dockerfile):

```
FROM mysql:5.7
# Copy SQL dump to initialization folder - This will
Import the SQL files to DB for the first time.
COPY ./init.sql /docker-entrypoint-initdb.d/
```

- **Docker Compose File** (/in-root-repo):

```
version: "3.8"

services:
  mysql_db:
    build: ./db/
    container_name: db_orthosecure
```

```
    restart: always
    ports:
      - "3306:3306"
    env_file:
      - ".env"
    volumes:
      - mysql_data:/var/lib/mysql
      - ./db/init.sql:/docker-entrypoint-initdb.d/init.sql
      - ./db/my.cnf:/etc/mysql/my.cnf
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "-h", "localhost",
"-u", "${MYSQL_USER}", "-
p${MYSQL_PASSWORD}"]
      interval: 10s
      timeout: 5s
      retries: 5
    networks:
      - orthosecure_network

  phpmyadmin:
    container_name: pma_orthosecure
    image: phpmyadmin/phpmyadmin
    restart: always
    ports:
      - "8080:80"
    environment:
      PMA_HOST: mysql_db
      MYSQL_ROOT_PASSWORD:
${MYSQL_ROOT_PASSWORD}
    depends_on:
      - mysql_db
    networks:
      - orthosecure_network

  orthosecure-app:
    build: ./app/
    container_name: app_orthosecure
    env_file:
      - ".env"
    ports:
      - "5000:5000"
    command: sh -c "./wait-for.sh mysql_db python
main.py"
    depends_on:
      mysql_db:
        condition: service_healthy
    networks:
      - orthosecure_network

volumes:
  mysql_data:

networks:
  orthosecure_network:
```

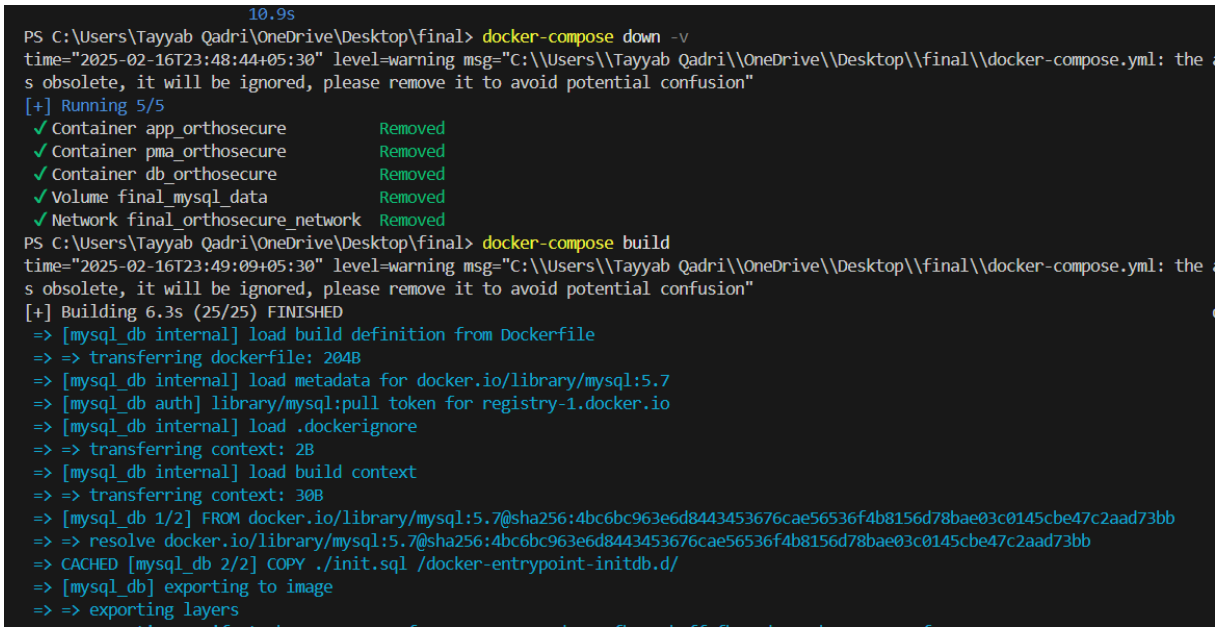
2. **Run a few Docker commands Locally to see if everything is correct**:

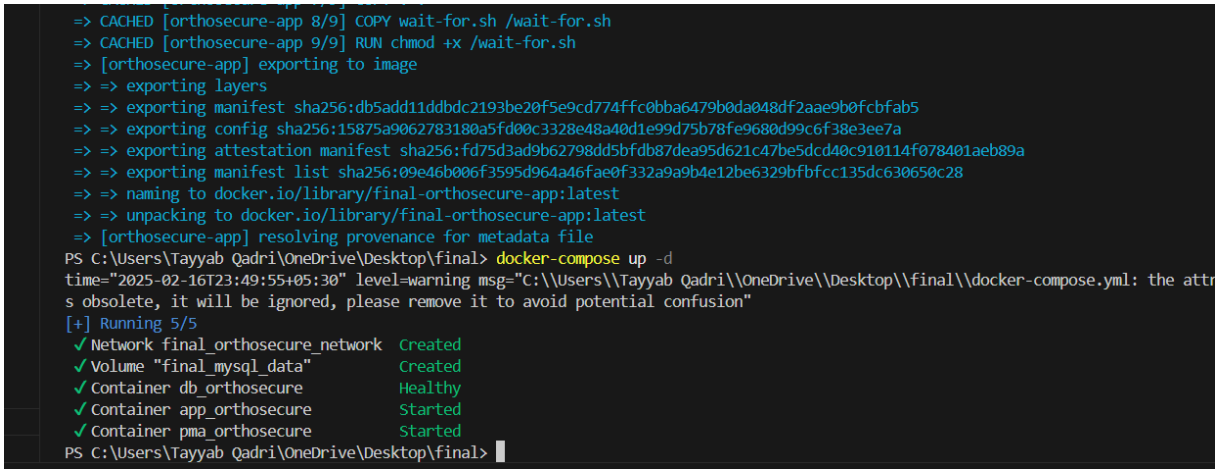
1. Stop and Remove Containers and Volumes:
```
**docker-compose down -v**
```
docker-compose down: Stops and removes all containers defined in docker-compose.yml.
-v: Also removes any associated volumes. This ensures that all data is removed, including the MySQL data volume.

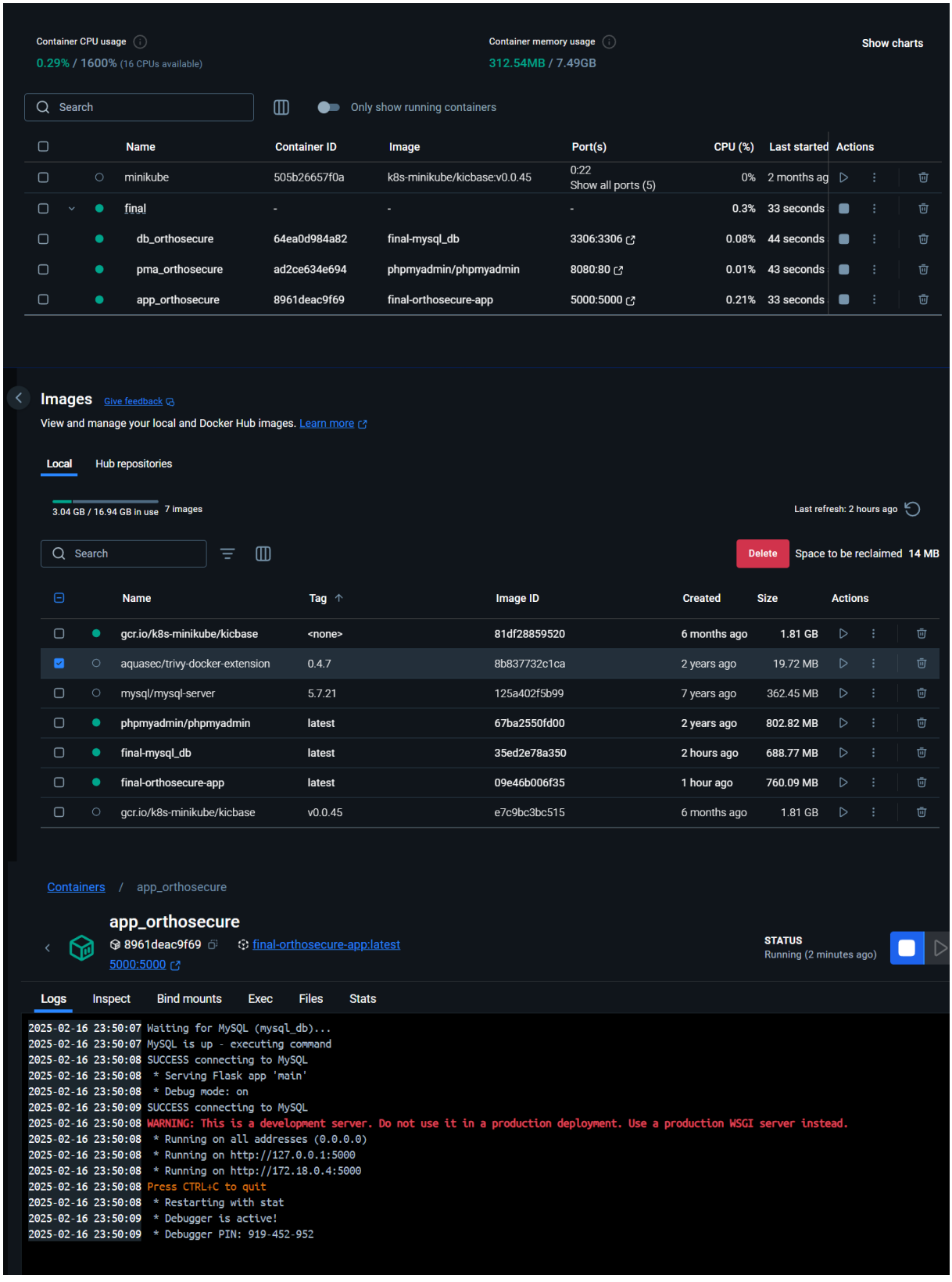
2. Rebuild Images:
```
**docker-compose build**
```
This will rebuild all images defined in your docker-compose.yml, including the MySQL image.

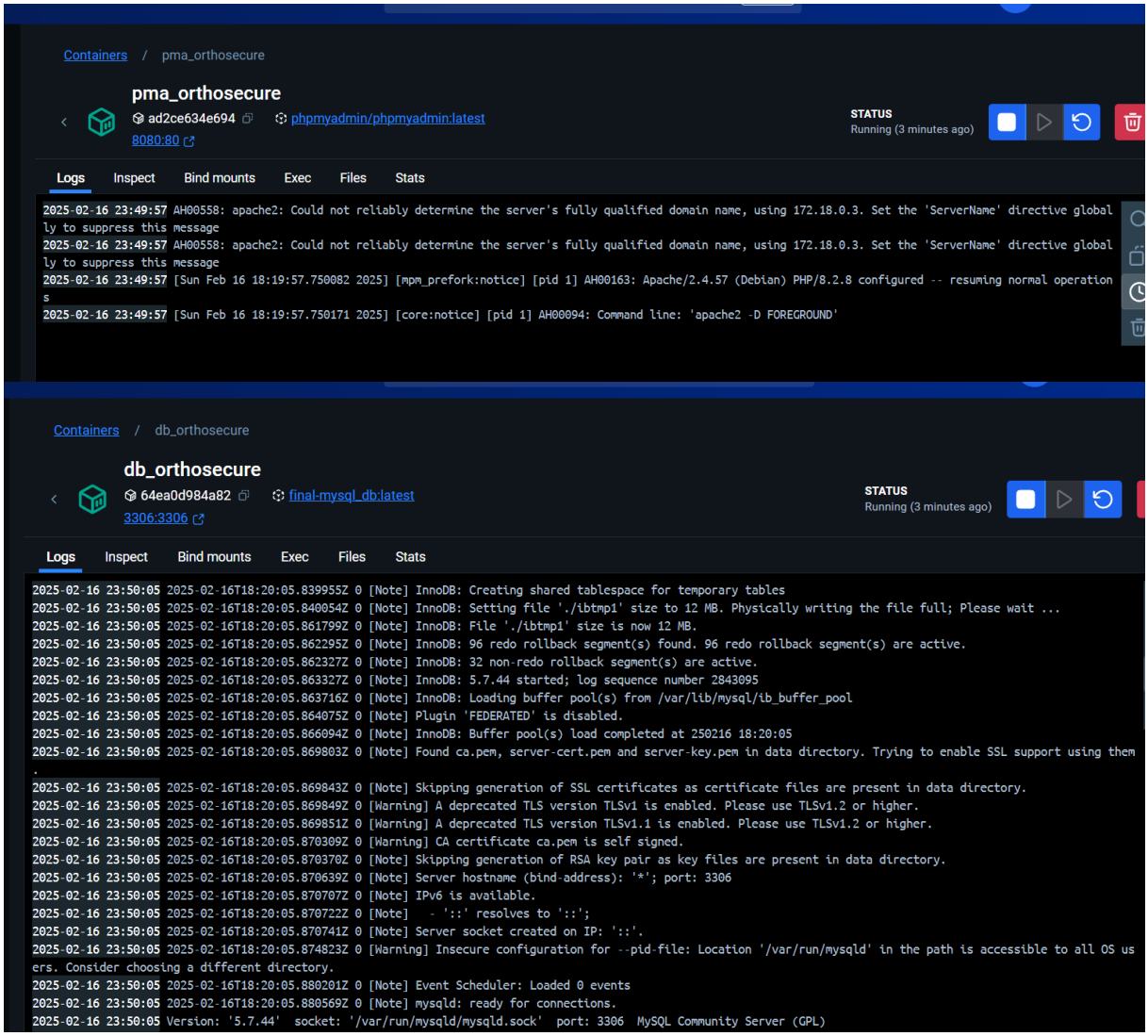
3. Start the Services:
```
**docker-compose up -d**
```
This will start all services defined in your docker-compose.yml with a clean slate

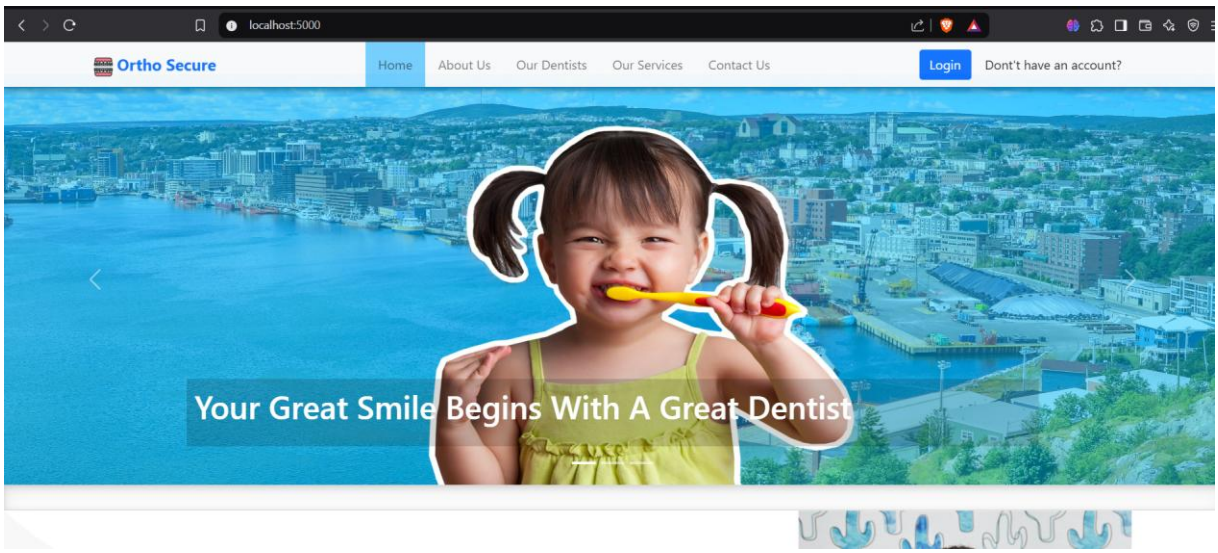
**Result:**




Container CPU usage
0.29% / 1600% (16 CPUs available)
Container memory usage
312.54MB / 7.49GB
Show charts
Search
Only show running containers
Name
Container ID
Image
Port(s)
CPU (%)
Last started
Actions
minikube
505b26657f0a
k8s-minikube/kicbase:v0.0.45
0:22
Show all ports (5)
0%
2 months ag
final
0.3%
33 seconds
db_orthosecure
64ea0d984a82
final-mysql_db
3306:3306
0.08%
44 seconds
pma_orthosecure
ad2ce634e694
phpmyadmin/phpmyadmin
8080:80
0.01%
43 seconds
app_orthosecure
8961deac9f69
final-orthosecure-app
5000:5000
0.21%
33 seconds
Images
Give feedback
View and manage your local and Docker Hub images. Learn more
Local
Hub repositories
3.04 GB / 16.94 GB in use
7 images
Last refresh: 2 hours ago
Search
Delete
Space to be reclaimed 14 MB
Name
Tag
Image ID
Created
Size
Actions
gcr.io/k8s-minikube/kicbase
<none>
81df28859520
6 months ago
1.81 GB
aquasec/trivy-docker-extension
0.4.7
8b837732c1ca
2 years ago
19.72 MB
mysql/mysql-server
5.7.21
125a402f5b99
7 years ago
362.45 MB
phpmyadmin/phpmyadmin
latest
67ba2550fd00
2 years ago
802.82 MB
final-mysql_db
latest
35ed2e78a350
2 hours ago
688.77 MB
final-orthosecure-app
latest
09e46b006f35
1 hour ago
760.09 MB
gcr.io/k8s-minikube/kicbase
v0.0.45
e7c9bc3bc515
6 months ago
1.81 GB
Containers / app_orthosecure
app_orthosecure
8961deac9f69
final-orthosecure-app:latest
5000:5000
STATUS
Running (2 minutes ago)
Logs
Inspect
Bind mounts
Exec
Files
Stats
2025-02-16 23:50:07 Waiting for MySQL (mysql_db)...
2025-02-16 23:50:07 MySQL is up - executing command
2025-02-16 23:50:08 SUCCESS connecting to MySQL
2025-02-16 23:50:08 * Serving Flask app 'main'
2025-02-16 23:50:08 * Debug mode: on
2025-02-16 23:50:09 SUCCESS connecting to MySQL
2025-02-16 23:50:08 WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
2025-02-16 23:50:08 * Running on all addresses (0.0.0.0)
2025-02-16 23:50:08 * Running on http://127.0.0.1:5000
2025-02-16 23:50:08 * Running on http://172.18.0.4:5000
2025-02-16 23:50:08 Press CTRL+C to quit
2025-02-16 23:50:08 * Restarting with stat
2025-02-16 23:50:09 * Debugger is active!
2025-02-16 23:50:09 * Debugger PIN: 919-452-952




**We Run the Docker Containers Locally & Cross-Check, if everything is fine we now proceed to deploy to IBM Cloud Container Registry**

**Step 2: Push Docker Images to IBM Cloud Container Registry**

1. Tag the images:

   Our docker-compose.yml has three services:
   - mysql_db → **MySQL Database**
   - orthosecure-app → **Our Actual Python App**
   - phpmyadmin → **PHPMyAdmin**

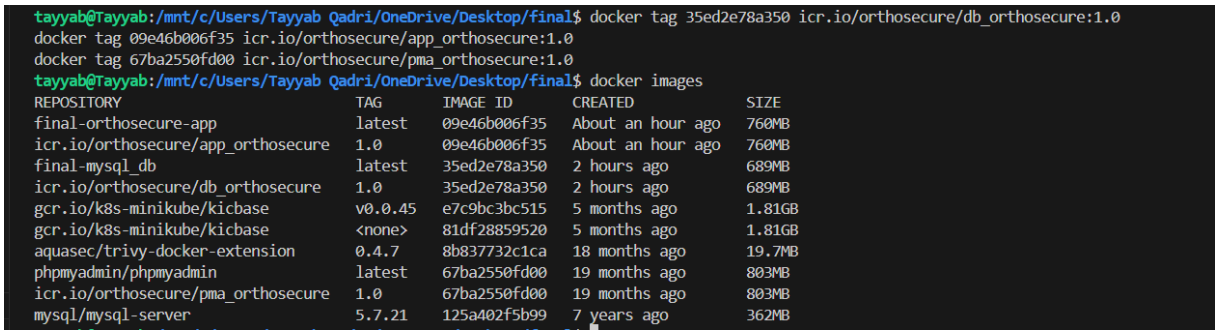
   Find your **IBM Cloud Region first**:

   **ibmcloud cr region** and then tag the images --- >

   docker tag <IMAGE_NAME> icr.io/orthosecure/db_orthosecure:1.0

   docker tag final-mysql_db icr.io/orthosecure/db_orthosecure:1.0
   docker tag final-orthosecure-app icr.io/orthosecure/app_orthosecure:1.0
   docker tag phpmyadmin/phpmyadmin icr.io/orthosecure/pma_orthosecure:1.0

   After Applying the Tags verify to proceed:



```
tayyab@Tayyab:/mnt/c/Users/Tayyab Qadri/OneDrive/Desktop/final$ docker tag 35ed2e78a350 icr.io/orthosecure/db_orthosecure:1.0
docker tag 09e46b006f35 icr.io/orthosecure/app_orthosecure:1.0
docker tag 67ba2550fd00 icr.io/orthosecure/pma_orthosecure:1.0
tayyab@Tayyab:/mnt/c/Users/Tayyab Qadri/OneDrive/Desktop/final$ docker images
REPOSITORY                          TAG       IMAGE ID       CREATED             SIZE
final-orthosecure-app               latest    09e46b006f35   About an hour ago   760MB
icr.io/orthosecure/app_orthosecure  1.0       09e46b006f35   About an hour ago   760MB
final-mysql_db                      latest    35ed2e78a350   2 hours ago         689MB
icr.io/orthosecure/db_orthosecure   1.0       35ed2e78a350   2 hours ago         689MB
gcr.io/k8s-minikube/kicbase         v0.0.45   e7c9bc3bc515   5 months ago        1.81GB
gcr.io/k8s-minikube/kicbase         <none>    81df28859520   5 months ago        1.81GB
aquasec/trivy-docker-extension      0.4.7     8b837732c1ca   18 months ago       19.7MB
phpmyadmin/phpmyadmin               latest    67ba2550fd00   19 months ago       803MB
icr.io/orthosecure/pma_orthosecure  1.0       67ba2550fd00   19 months ago       803MB
mysql/mysql-server                  5.7.21    125a402f5b99   7 years ago         362MB
```

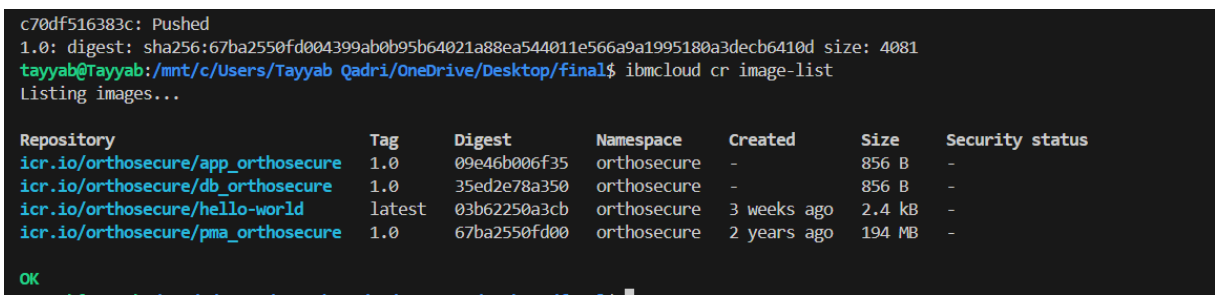
2. Push the images:

   docker push icr.io/orthosecure/db_orthosecure:1.0
   docker push icr.io/orthosecure/app_orthosecure:1.0
   docker push icr.io/orthosecure/pma_orthosecure:1.0

3. Verify Uploaded Images:

   After pushing the images, verify that they have been uploaded successfully by running:

   **ibmcloud cr image-list**



```
c70df516383c: Pushed
1.0: digest: sha256:67ba2550fd004399ab0b95b64021a88ea544011e566a9a1995180a3decb6410d size: 4081
tayyab@Tayyab:/mnt/c/Users/Tayyab Qadri/OneDrive/Desktop/final$ ibmcloud cr image-list
Listing images...

Repository                          Tag      Digest         Namespace     Created       Size     Security status
icr.io/orthosecure/app_orthosecure  1.0      09e46b006f35   orthosecure   -             856 B    -
icr.io/orthosecure/db_orthosecure   1.0      35ed2e78a350   orthosecure   -             856 B    -
icr.io/orthosecure/hello-world      latest   03b62250a3cb   orthosecure   3 weeks ago   2.4 kB   -
icr.io/orthosecure/pma_orthosecure  1.0      67ba2550fd00   orthosecure   2 years ago   194 MB   -

OK
```

## SECTION 2: TESTING THE SOLUTION

### Step 1: Set Up Minikube and Deploy Applications

1. **Start Minikube**:

minikube start

**Our Kubernetes Structure and it's Implementation is extended so here is a tree like structure for this project so that it grasps out all the directory**

```
tayyab@Tayyab:/mnt/c/Users/Tayyab Qadri/OneDrive/Documents/OrthoSecure/kubernetes$ ls -R
.:
configmaps  deployments  kubernetes.txt  policies  pvc  secrets  service-accounts  services  tests

./configmaps:
app-config.yaml  mysql-config.yaml

./deployments:
app-deployment.yaml  dummy.yaml  mysql-deployment.yaml  phpmyadmin-deployment.yaml  security-context.yaml

./policies:
admission-controllers  namespaces  network-policy.yaml  pod-security-policy.yaml  rbac

./policies/admission-controllers:
gatekeeper-constraints.yaml

./policies/namespaces:
secure-namespace.yaml

./policies/rbac:
role-restricted-deployer.yaml  rolebinding-restricted.yaml

./pvc:
mysql-pvc.yaml

./secrets:
mysql-secrets.yaml

./service-accounts:
app-service-account.yaml

./services:
app-service.yaml  mysql-service.yaml  phpmyadmin-service.yaml

./tests:
test-pod-security.yaml  test-rbac-pod.yaml
tayyab@Tayyab:/mnt/c/Users/Tayyab Qadri/OneDrive/Documents/OrthoSecure/kubernetes$
```

2. **Create Kubernetes Deployment and Service YAML Files**:

   - **Frontend Deployment** (app-deployment.yaml):

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app-deployment
  labels:
    app: app
    environment: production # extra label
spec:
  replicas: 2
  selector:
    matchLabels:
```

```
    app: app
 template:
  metadata:
    labels:
      app: app
      environment: production # extra label beacuse we are in productions stage now
  spec:
    containers:
      - name: app-orthosecure
        image: nidithvs/app_orthosecure:latest
        ports:
          - containerPort: 5000
        envFrom:
          - configMapRef:
              name: app-config
        command: ["sh", "-c", "./wait-for.sh mysql-deployment python main.py"]
        resources:
          requests:
            memory: "256Mi"
            cpu: "250m"
          limits:
            memory: "512Mi"
            cpu: "500m"
```

- **Backend Deployment** (backend-deployment.yaml):

```
  apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deployment
  labels:
    app: mysql-orthosecure
    environment: production
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql-orthosecure
  template:
    metadata:
      labels:
        app: mysql-orthosecure
        environment: production
    spec:
      containers:
        - name: db-orthosecure
          image: nidithvs/db_orthosecure:latest
          ports:
            - containerPort: 3306
          envFrom:
            - configMapRef:
                name: mysql-config
            - secretRef:
                name: mysql-secrets
```

```
        volumeMounts:
          - name: mysql-storage
            mountPath: /var/lib/mysql
        resources:
          limits:
            memory: "512Mi"
            cpu: "500m"
          requests:
            memory: "256Mi"
            cpu: "250m"
        readinessProbe:
          exec:
            command: ["sh", "-c", "mysqladmin ping -h localhost -u $MYSQL_USER -
p$MYSQL_PASSWORD"]
          initialDelaySeconds: 10
          periodSeconds: 10
        livenessProbe:
          exec:
            command: ["sh", "-c", "mysqladmin ping -h localhost -u $MYSQL_USER -
p$MYSQL_PASSWORD"]
          initialDelaySeconds: 30
          periodSeconds: 30
      volumes:
        - name: mysql-storage
          persistentVolumeClaim:
            claimName: mysql-pvc

# Notes:

# Image: Ensure that your-registry/mysql_db:latest is accessible from your Kubernetes cluster. This
typically means pushing your built Docker image to a container registry like Docker Hub, Google
Container Registry, or a private registry.
# Probes: Kubernetes doesn't support environment variable interpolation in probe command arrays
directly. By using sh -c, we ensure that shell expansion occurs.
# Security: Ensure your MySQL user has the necessary privileges and that passwords are secure.
```

- ## Apply YAML Files:

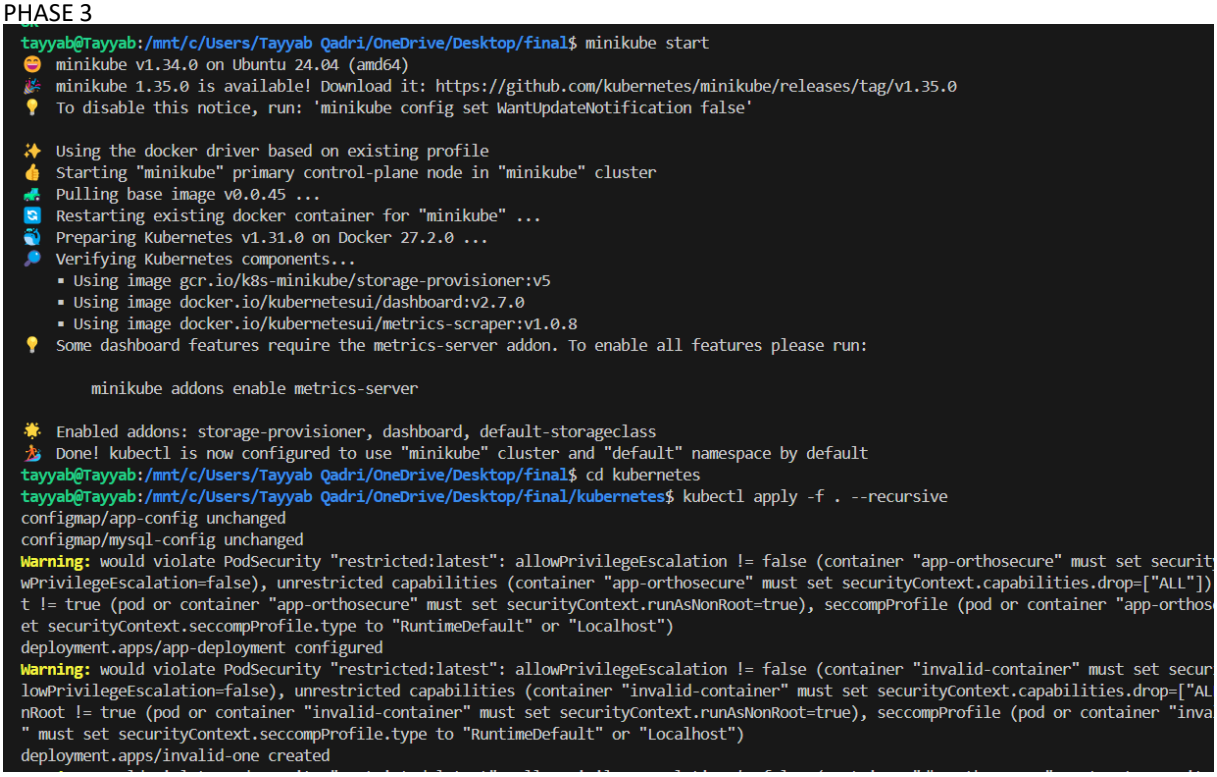
kubectl apply -f <Yaml file name>

Single Command to Apply All Manifests, Run this in **kubernetes** directory:

kubectl apply -f . –recursive

**What this does?**

- The . refers to **the current directory**
- --recursive ensures that **all subdirectories** are included




## Step 2: Verify Deployments

1. Check running pods:

**kubectl get pods**

2. Check services:

**kubectl get svc**

3. Check Deployments:

**Kubectl get Deployments**

4. Access applications:
    - Use the Minikube service IP or tunnel to expose services.

# Result is in Next Page

PHASE 3

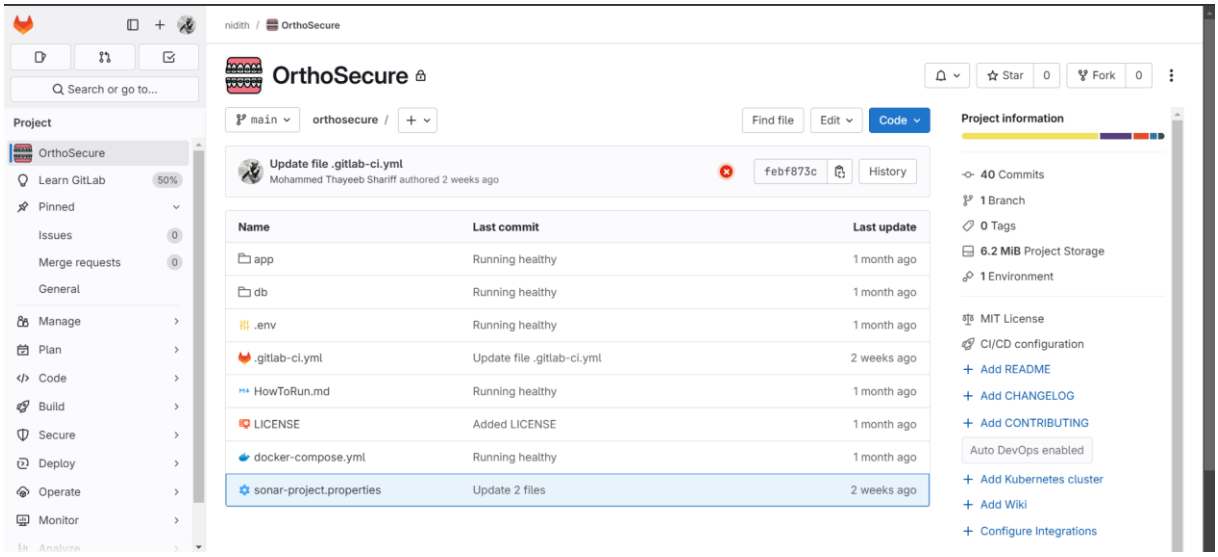
```
tayyab@Tayyab:/mnt/c/Users/Tayyab Qadri/OneDrive/Desktop/final/kubernetes$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS        AGE
app-deployment-6976db76c-4576s        1/1     Running   2 (7m42s ago)   49d
app-deployment-6976db76c-d5692        1/1     Running   2 (7m42s ago)   49d
grafana-5cf7b7b89f-7db4v              1/1     Running   2 (48d ago)     49d
mysql-deployment-7d686dd697-nsjb6     1/1     Running   2 (48d ago)     49d
phpmyadmin-deployment-67cfb5577f-x2crd   1/1     Running   2 (48d ago)     49d
tayyab@Tayyab:/mnt/c/Users/Tayyab Qadri/OneDrive/Desktop/final/kubernetes$ kubectl get svc
\NAME                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
grafana               ClusterIP   10.109.147.230   <none>        80/TCP           49d
kubernetes            ClusterIP   10.96.0.1        <none>        443/TCP          49d
mysql-service         ClusterIP   10.108.219.122   <none>        3306/TCP         6m39s
orthosecure-service   NodePort    10.102.229.81    <none>        5000:30050/TCP   6m39s
phpmyadmin-service    NodePort    10.97.136.210    <none>        80:30080/TCP     6m39s
trivy                 ClusterIP   10.103.31.85     <none>        4954/TCP         49d
tayyab@Tayyab:/mnt/c/Users/Tayyab Qadri/OneDrive/Desktop/final/kubernetes$ kubectl get deployments
NAME                 READY   UP-TO-DATE   AVAILABLE   AGE
app-deployment       2/2     0            2           49d
grafana              1/1     1            1           49d
invalid-one          0/1     0            0           6m43s
mysql-deployment     1/1     0            1           49d
phpmyadmin-deployment   1/1     0            1           49d
```

## Step 3: Testing CI/CD Integration

1. Set up GitHub Actions with a CI/CD pipeline YAML file.

2. Automate build, test, and deployment stages using Minikube and IBM Cloud CLI.

# Reuslts:




---

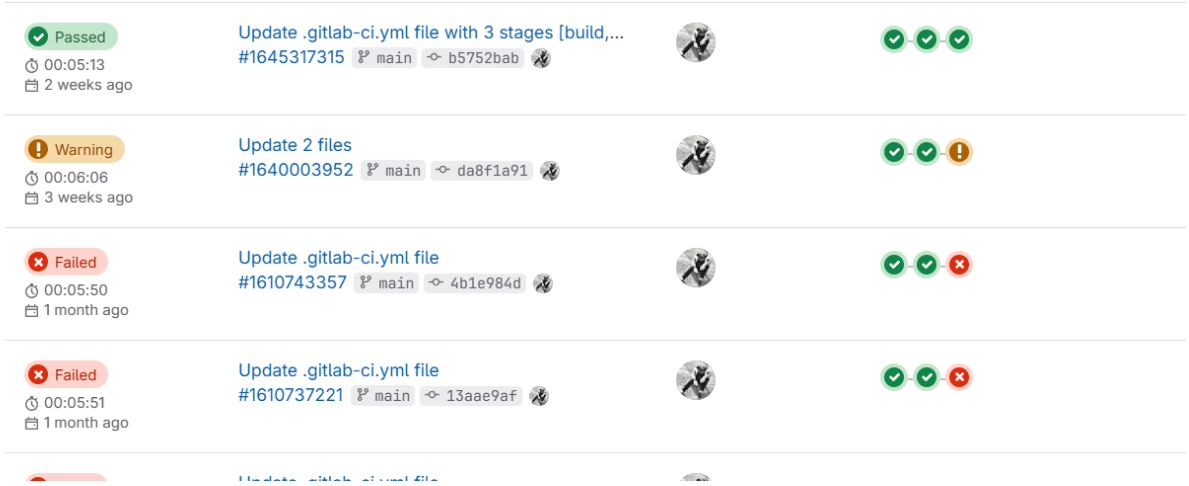
# GitLab CI/CD File

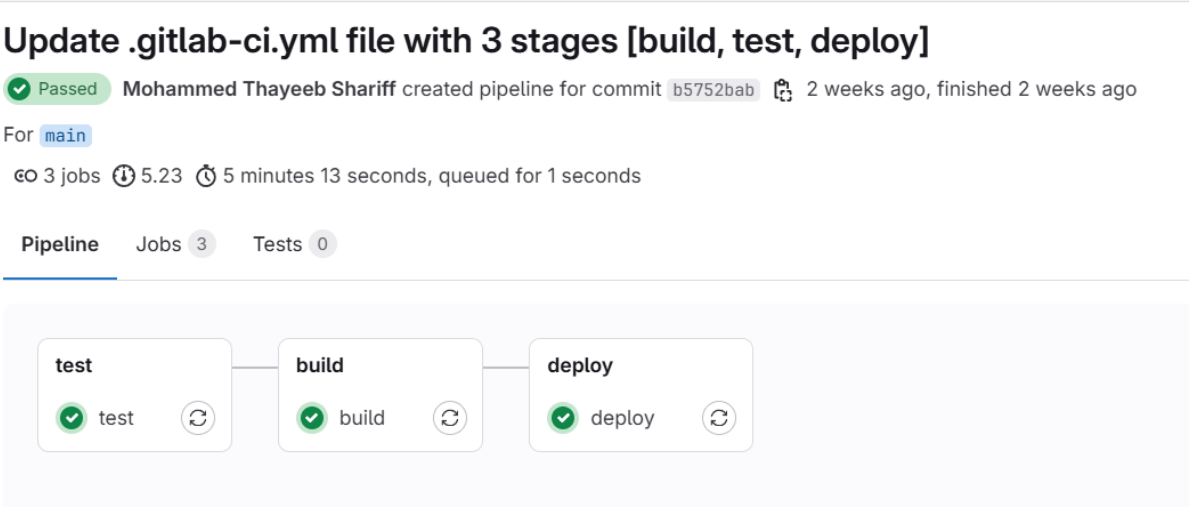
```
# Define stages
# stages:
#   - test
#   - build
#   - deploy
#   - sonarqube-check
```

```
# # Global variables
# variables:
#   DOCKER_DRIVER: overlay2
#   DOCKER_HOST: tcp://docker:2375/
#   COMPOSE_PROJECT_NAME: orthosecure
#   SONAR_USER_HOME: "${CI_PROJECT_DIR}/.sonar" # Location for SonarQube analysis cache
#   GIT_DEPTH: "0"                      # Fetch all branches for SonarQube analysis

# # Docker service configuration
# services:
#   - docker:dind

# # Install Docker Compose before any job
# before_script:
#   - apt-get update && apt-get install -y curl
#   - curl -L "https://github.com/docker/compose/releases/download/v2.20.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
#   - chmod +x /usr/local/bin/docker-compose
#   - docker-compose --version


# # Test stage
# test:
#   stage: test
#   script:
#     - docker-compose down -v  # This ensures that all the data has been removed, including the MySQL data volume.
#   allow_failure: false

# # Build stage
# build:
#   stage: build
#   script:
#     - docker-compose -f docker-compose.yml build
#   artifacts:
#     paths:
#       - ./db/
#       - ./app/
#     expire_in: 1 week

# # Deploy stage
# deploy:
#   stage: deploy
#   only:
#     - main
#   script:
#     - docker-compose -f docker-compose.yml up -d

# # SonarQube analysis stage
# sonarqube-check:
#   stage: sonarqube-check
#   image:
#     name: sonarsource/sonar-scanner-cli:11
#     entrypoint: [""]
#   script:
```

PHASE 3

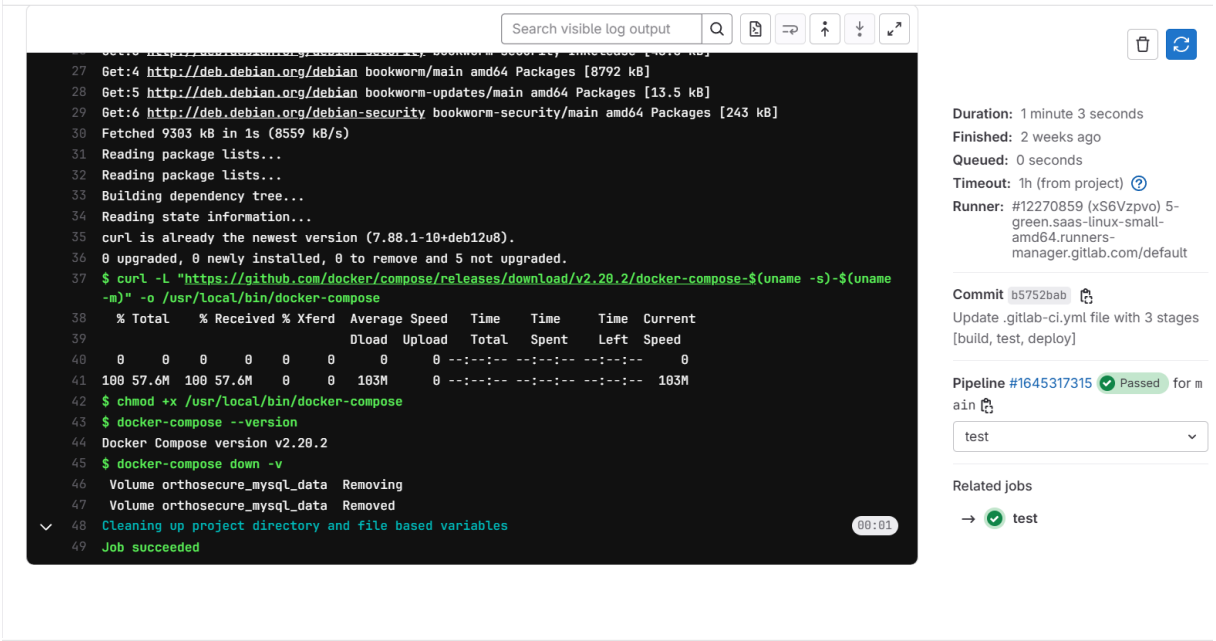
```
#     - sonar-scanner -Dsonar.host.url="${SONAR_HOST_URL}"
#   allow_failure: true
#   rules:
#     - if: $CI_PIPELINE_SOURCE == 'merge_request_event'
#     - if: $CI_COMMIT_BRANCH == 'master'
#     - if: $CI_COMMIT_BRANCH == 'main'
#     - if: $CI_COMMIT_BRANCH == 'develop'
```

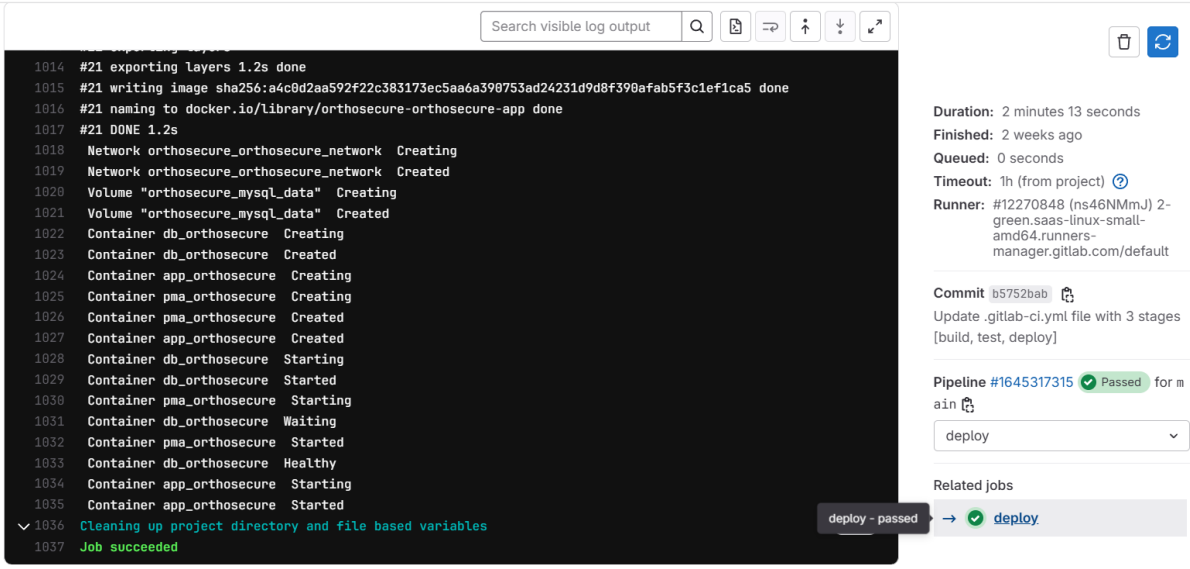


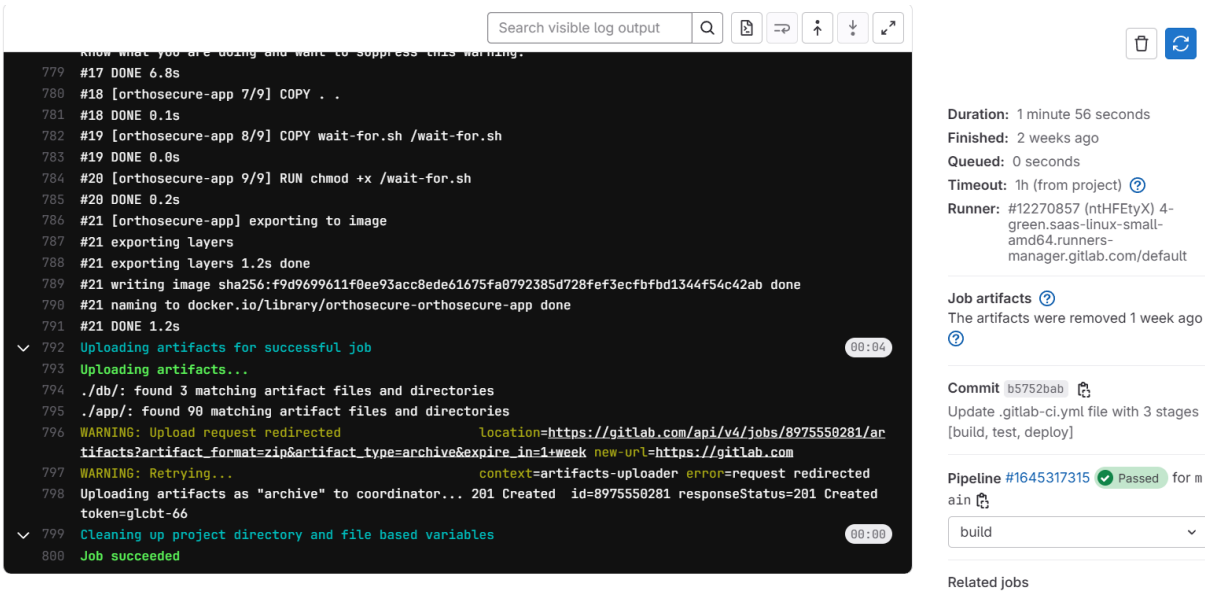




DEVOPS ENGINEER

PHASE 3




DEVOPS ENGINEER

## SECTION 3: FUTURE IMPROVEMENTS

## GOAL-ORIENTED WITH STRATEGIES AND TECH STACK:

1. **Plan 1: Advanced Threat Intelligence**
    - **Goal**: Utilize AI/ML for predictive threat detection and proactive defense.
    - **Strategy**: Integrate AI/ML models into monitoring tools like Prometheus and ELK Stack to identify anomalies in real time.
    - **Tech Stack**: TensorFlow, PyTorch, Prometheus, ELK Stack.
2. **Plan 2: Security-as-Code**
    - **Goal**: Codify security policies for repeatable and scalable implementations.
    - **Strategy**: Create Infrastructure as Code (IaC) templates using tools like Terraform with embedded security standards.
    - **Tech Stack**: Terraform, AWS CloudFormation, HashiCorp Vault.
3. **Plan 3: Multi-Cloud Security Expansion**
    - **Goal**: Ensure consistent security across hybrid and multi-cloud environments.
    - **Strategy**: Leverage cloud-native security tools from AWS, Azure, and GCP to establish a unified multi-cloud security posture.
    - **Tech Stack**: Kubernetes, Istio, Cloud-native security tools (AWS Shield, Azure Security Center).
4. **Plan 4: Open-Source DevSecOps Framework**
    - **Goal**: Foster innovation and broad adoption by releasing the framework as open-source.
    - **Strategy**: Host the framework on GitHub, document it comprehensively, and build a community for contributions.
    - **Tech Stack**: GitHub, Markdown for documentation, Swagger for API documentation.